ENTRY
.
.
.
.

Test   X
BC   EQ,Jump1   Branch B1
.
.
.
.

Test   X
BC   EQ,Jump2   Branch B2
.
.

Touch A   Prefetch A
.
.

Load   R5,A   Use   A
.
Return
Jump1 *
.
.

Test   X
BC   EQ,Jump3   Branch B3
.

Touch C   Prefetch C
.
.

Load   R5,C   Use   C
.
Return
Jump2 *
.
.

Touch B   Prefetch B
.
.

Load   B   Use   B
.
Return
Jump3 *
.

Touch D   Prefetch D
.
.

Load   D   Use   D
.
Return
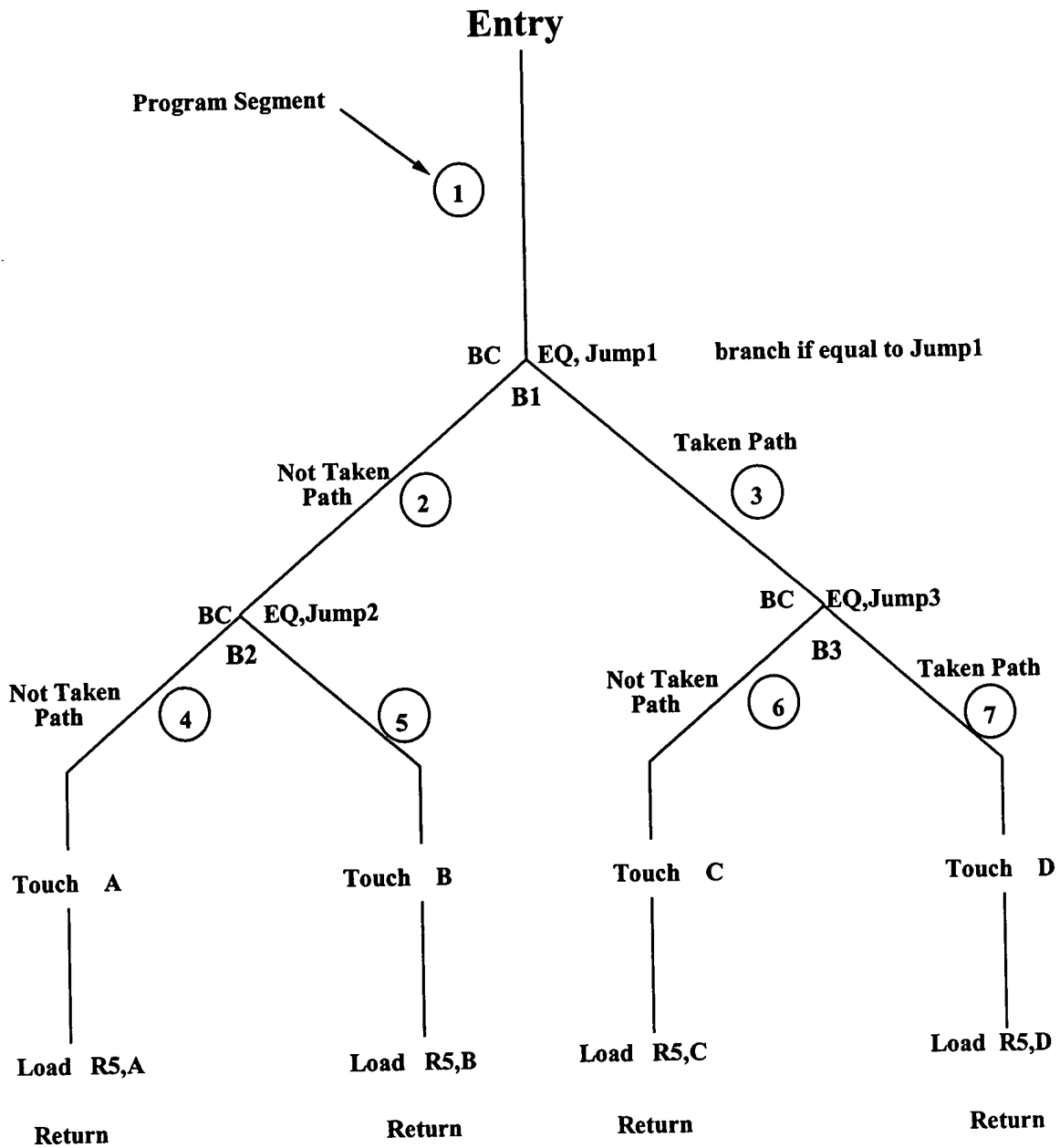
Program Segment

1

2

4

3

6

5

7

Figure   1

Control Flow Graph Of Program

Entry

Program Segment ①

BC ╲ EQ, Jump1   branch if equal to Jump1

B1

Not Taken Path ②   Taken Path ③

BC ╲ EQ,Jump2   BC ╲ EQ,Jump3

B2   **B3**

Not Taken Path ④   ⑤   Not Taken Path ⑥   Taken Path ⑦

Touch   A   Touch   B   Touch   C   Touch   D

Load   R5,A   Load   R5,B   Load   R5,C   Load   R5,D

Return   Return   Return   Return

Figure 2

Control Flow Graph Of Program

Entry

Program Segment

(1)

Touch A
Touch B   Prefetch A, B, C, and D
Touch C
Touch D

BC EQ, Jump1   branch if equal to Jump1

B1

Not Taken
Path

(2)

Taken Path

(3)

BC EQ,Jump2

(4) B2 (5)

Not Taken
Path   Taken Path

BC EQ,Jump3

(6) B3 (7)

Not Taken
Path   Taken Path

Load R5,A   Load R5,B   Load R5,C   Load R5,D

Return   Return   Return   Return

Figure 3

TOUCH INSTRUCTION FORMAT

50

| OPCODE 51 | BRANCH MASK 52 | PREFETCH ADDRESS 53 |

N Sub−Fields

Sub−Field 1 = Execution Action of First Branch After Touch Instruction

Sub−Field 2 = Execution Action of Second Branch After Touch Instruction
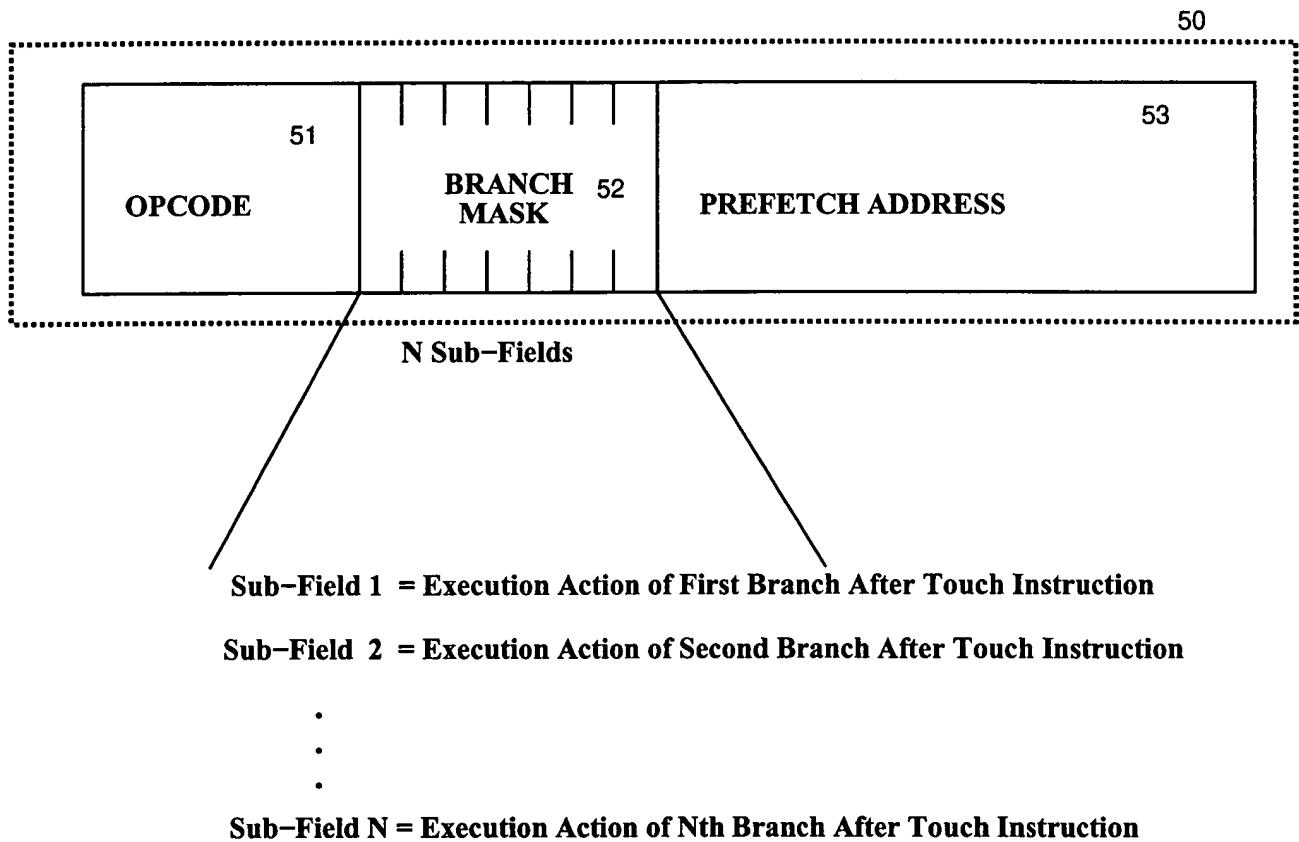
.
.
.

Sub−Field N = Execution Action of Nth Branch After Touch Instruction

The Branch Action Can Be One Of The Following Values:
T = The Branch Is Taken
N = The Branch Is Not−Taken
D = The Branch Can Either Be Taken Or Not−Taken, 'Don't Care'

Prefetch Address Can Denote A Base Register + Displacment or
A Relative Offset From the Touch Instruction

Figure 4

Memory 5

Stores and Prefetches From Execution Units

Instruction Fetch

Cache 10

Operand Return

Instruction return

Instruction Buffers

Operand Requests

15

Instruction Fetch Logic 20

Decoder 25

Touch Detection

Instruction Fetch Sequence Updates

Instruction Fetch Sequence Updates

200 Touch Exec Unit

250 General Exec Unit

300 Branch Exec Unit

Branch Results

400

Branch History Queue

985

Execution Units

995

910

960

30

901

975

Branch Prediciton Mechanism

965

100 Pending Branch Prediction Logic

950

970

Branch Updates

955

Branch Prediction Updates

Branch History Mask Updates

990

Figure 5

Pending Branch Prediction Logic

From Decoder Branch and Touch Instruction Information

100

901

102

Branch | **Branch or Touch Instruction** | Touch

To Branch Execution Unit

975

101

Branch Decode Logic

150

Touch Instruction Decode Logic

Update Info

125

Pending Branch Prediction Queue (PBPQ)

Data

Data

To Touch Execution Unit

960

965

Update Information

Update Information

To Decoder

175

Branch Execute Logic

190

Branch Prediction Hit Logic

970

To Inst–Fetch Logic

950 | **From Branch Execution Unit**

955 | **From Branch Prediciton Mechanism**

Figure 6

BRANCH HISTORY QUEUE

From Branch
E−Unit

400

New Branches
Enter From
Top

985

Branch Address Information Field 410	Branch History Mask Field 420
411 Branch Address 1	421 $--\cdots -- X_n$
Branch Address 2	$--\cdots -X_{n-1}X_n$
Branch Address 3	$--\cdots X_{n-2}X_{n-1}X_n$
Branch Address N−1	$-X_2\cdots X_{n-2}X_{n-1}X_n$
Branch Address N	$X_1X_2\cdots X_{n-2}X_{n-1}X_n$

For Each New Branch Entering
The Queue Four Events Occur

1. All Branch Addresses And
   Branch History Information
   Are Pushed Down One Position
   In The Queue.

2. All Branch History Masks
   Are Shifted Left One Bit.

3. The New Branch Address Is
   Placed In The First Position
   Of The Queue, Branch−Address−1.

4. The Branch Action
   (Taken/Not−Taken) Of The
   New Branch Is Placed As The
   Right Most Bit To All
   Branch History Masks.

When Complete, The Branch History Mask
Is N Bits Wide

1 = Branch Was Taken
0 = Branch Was Not−Taken

Old Branches Leave From
Bottom And Save Branch
History Mask In BHT

990 To Branch Prediction Mechanism

Figure 7

PENDING BRANCH PREDICTION QUEUE

Branch Decode Logic

To Touch Execution Unit

960

901

125

From BHT

	Branch Address Information Field 126	Target Address Information Field 128	Branch History 130 Mask Field	Valid Bit 132 Field	To Branch Decode Logic
	Branch Address 1 ^{140}	Target Address 1 ^{142}	$X_1X_2X_3 \cdots X_{n-1}X_n$ ^{144}	V ^{146}	Touch Decode Logic
	Branch Address 2	Target Address 2	$X_1X_2X_3 \cdots X_{n-1}X_n$	V	Branch Prediction Hit Logic
	Branch Address 3	Target Address 3	$X_1X_2X_3 \cdots X_{n-1}X_n$	V	
	Branch Address 4	Target Address 4	$X_1X_2X_3 \cdots X_{n-1}X_n$	V	

Branch History Mask Records The Branch Action Of Last N Branches
          1 = Taken
          0 = Not–Taken

Branch Address Information

          Address Of Predicted Branch Detected During Branch Prediction

Target Address Information

          Predicted Target Address of Branch Detected During Branch Prediction

Valid Bit
          1 = Valid
          0 = Invalid

Figure 8

Branch Prediciton Mechanism Hit Logic

From Branch Prediction Mechanism   955

Branch Address	Target Address	Mask

195

191

First Position
Valid In
PBPQ

Valid

Not Valid

192

Find First
Available
Position Of
PBPQ.

Place Branch
Prediciton
Information In
First Available
Position

194

Place Branch
Prediciton
Information
In First Position
Of PBPQ

PBPQ 125

140	142	144	146
Br Addr	Tgt Addr	Mask	V

Figure 9

Branch Decode Logic

101

From Decode Unit
Branch Instruction
Information

901

108
Not Valid Detect

Not Valid

103
First Position of PBPQ Valid

Branch Address Of First Entry

109
Check Condition Code

PBPQ 125

140	142	144	146
Br Addr	Tgt Addr	Mask	V

Pending Branch Prediction Queue

Conditional

110
Set Decode PC, Next Sequential

Unconditional

Set Decode PC to Branch Target Address **112**

Valid

104
Valid Detect

105
Valid Detect

114
Send Branch Guess Info To Branch Execution Unit

Equal   905

106
Compare Branch Address

To Decode Unit

965

116
Check Branch Action, Taken Not-Taken

Compare

Not Equal   906

Purge

107
Branch Prediction Error

Purge PBPQ

Not Taken

Taken

PBPQ 125

140	142	144	146
Br Addr	Tgt Addr	Mask	V

Pending Branch Prediction Queue

118
Set Decode PC to Next Sequential Instruction Address

120
Reset Decode PC to Branch Target Address

111
Reset Instruction Fetch Address

Send Address To Instruction Fetch Logic Unit

Reset Decoder PC

To Branch Execution Unit

975

122
Send Branch Guess Info To Branch Execution Unit

124
Pop PBPQ. Remove First Position Of PBPQ.

970

To Instruction Fetch Logic Unit

Figure 10

Touch Instruction Decode Logic

From Decoder
Touch Instruction Information
And Branch Mask

150

910

912

140	142	144	146
Br Addr	Tgt Addr	Mask	V

151 First Position Of PBPQ Valid

Yes

No

161 Discard Instruction

152 Valid Detect

154 Valid Detect

156 Mask Compare Logic

Not Equal

Equal

160 Construct Touch Instruction Information Block

162 Discard Instruction

BMVB = 1 164

960

To Touch Instruction
Execution Unit

Figure 11

Branch Mask Compare Logic

Input From Touch Instruction Branch Mask

	T	N	D
Input From Branch History Mask **0**	No Match	Match	Match
1	Match	No Match	Match

{ 0 = Not Taken

1 = Taken

Figure 12

Touch Instruction Information Block Format   170

Branch Mask ¹⁷¹	Branch IID ¹⁷²	BMVB ¹⁷³	Execution Information ¹⁷⁴

Branch Mask = Branch Mask Contained In Touch Instruction

Branch IID = Instruction Identifier Of Last Branch Decoded

BMVB = Branch Mask Validation Bit

Execution Information = Address Of Item To Prefetch, ...

Figure 13

Branch Execute Logic

175

950

From Branch Execution
Unit

Branch
Guess

Branch
Action

Predicted
Branch
Target
Address

Actual
Branch
Target
Address

177

**Branch Action
Compare Logic**

Both
Not–Taken

Both
Taken

179

**Branch Address
Compare Logic**

925

927

929

931

Not Equal

933

Equal

180

No
Action

Mixed
Taken
and
Not–Taken

182

'AND'
Detect

188

No
Action

184

Purge PBPQ

To Inst
Fetch Logic

**Restart Instruction
Fetch Logic**

185

**Restart Decode
Logic**

186

970

965

To Decoder

Figure 14

Touch Instruction Execution Unit

960

From
Branch
Execution
Unit   200

Touch
Instruction
Execution
QUeue

262

995

Branch IID

Touch Information Block   170

Branch Mask 171	Branch IID 172	BMVM 173	Execution Information 174

205

Branch IDD Compare
Logic

Equal

210

BMVM
Equal
Valid

Invalid

Valid

Valid
Detect
211

Discard
Instruction

212

215

Execution Unit

Prefetch Request
To Cache

Figure 15